# 第6章 汇编语言

## 6.1 实方式执行环境

6.2 源程序和语句

6.3 操作数表示

6.4 伪指令语句和变量

6.5 段声明和段间转移

6.6 目标文件和段模式

6.7 宏

# 6.1 实方式执行环境

## 6.1.1 寄存器和指令集

## 6.1.2 存储器分段管理

## 6.1.3 16位存储器寻址方式

# 6.1.1 寄存器和指令集

## ➢寄存器

- ✓ 32位寄存器EAX、EBX、ECX、EDX、ESP、EBP、ESI和EDI；
  16位寄存器AX、BX、CX、DX、SP、BP、SI和DI；
  8位寄存器AH、AL、BH、BL、CH、CL、DH和DL。
- ✓ 段寄存器CS、DS、SS和ES，以及段寄存器FS和GS。寄存器CS含有当前代码段的段值，寄存器DS含有当前数据段的段值，寄存器SS含有当前堆栈段的段值。
- ✓ 实方式下指令指针寄存器EIP中的高16位必须是0，相当于只有低16位的IP起作用。
- ✓ 实方式下堆栈指针寄存器ESP中的高16位必须是0，相当于只有低16位的SP起作用。

# 6.1.1 寄存器和指令集

➢指令集

- ✓ 数据传送指令组
- ✓ 算术运算指令组
- ✓ 逻辑运算指令组
- ✓ 移位指令组
- ✓ 转移指令组
- ✓ 字符串操作指令组
- ✓ 位操作指令组
- ✓ 条件字节设置指令组
- ✓ 其他指令

# 6.1.2 存储器分段管理

## ➢存储器分段条件

- ✓ IA-32系列处理器的物理地址空间规模达到4G
- ✓ 实方式下可访问的物理地址空间只有1M

  00000H -- FFFFFH

- ✓ 实方式下每个逻辑段必须满足如下两个条件：

  第一，逻辑段的起始地址必须是16的倍数

  第二，逻辑段的最大长度为64K

- ✓ 存储段，既可以相连，也可以重叠

最初的**Intel 8086**处理器是**16**位的，
这两个条件是为了方便地计算**1M**空间中的**20**位地址。

# 6.1.2 存储器分段管理

## ➢物理地址计算

- ✓ 实方式下，由于段的起始地址必须是16的倍数，因此段起始地址有如下形式：

  bbbbbbbbbbbbbbbb0000

- ✓ 段起始地址采用十六进制可表示成XXXX0。这种20位的段起始地址，可压缩表示成16位的XXXX形式。把20位段起始地址的高16位XXXX称为**段值**。
- ✓ 段起始地址与段值的关系如下：

  段起始地址 = 段值×16

- ✓ 物理地址、段值和偏移之间有如下关系：

  物理地址 = 段值×16 + 偏移

# 6.1.2 存储器分段管理

## ➢物理地址计算

把20位段起始地址的高16位XXXX称为段值


分段的条件：

第一，逻辑段的起始地址必须是16的倍数

第二，逻辑段的最大长度为64K

# 6.1.2 存储器分段管理

## ✓ 地址计算示例

一些存储单元的逻辑地址和对应的物理地址如下所列，
左边是逻辑地址，右边是对应的物理地址

| | |
|---|---|
| 1234:3456 | 15796 |
| 1234:34A8 | 157E8 |
| FFF0:0000 | FFFF0 |

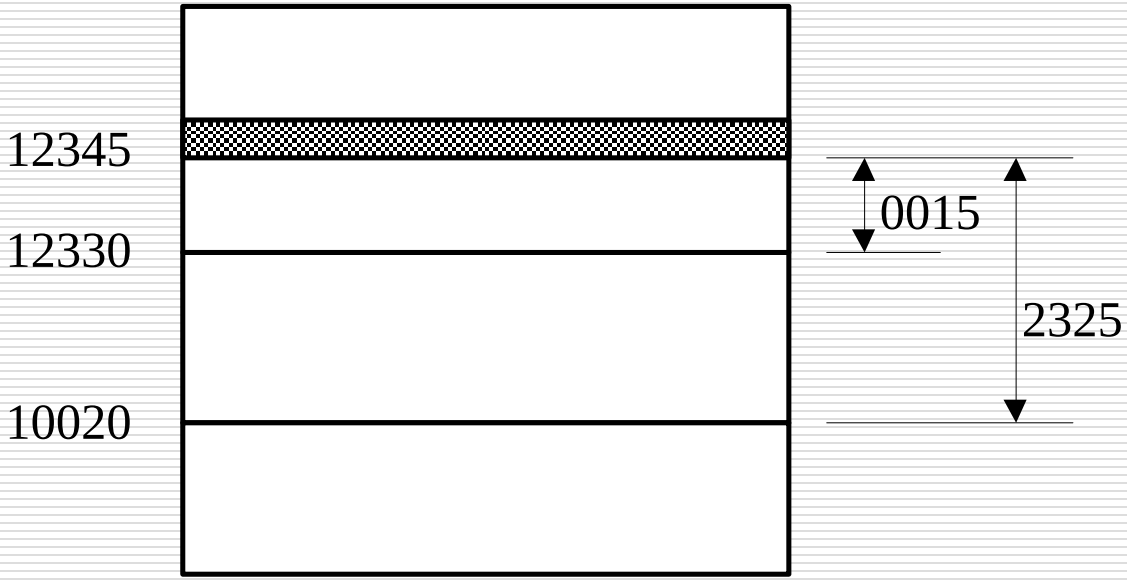
采用十六进制表示

# 6.1.2 存储器分段管理

## ✓ 地址计算示例

由于段可以重叠，一个物理地址可对应多个逻辑地址

```
1002:2325 = 12345
1233:0015 = 12345
```



# 6.1.2 存储器分段管理

## ➢段寄存器引用

- ✓ 实方式下，段寄存器中的内容是段值。
- ✓ 代码段寄存器CS给出当前代码段的段值，堆栈段寄存器SS给出当前堆栈段的段值，数据段寄存器DS给出当前缺省数据段的段值。附加段寄存器ES、FS、GS也可以给出其他数据段的段值。
- ✓ 每当需要产生一个20位的物理地址时，CPU会自动引用一个段寄存器获得段值，形成20位的段起始地址，再加上有效地址（偏移）。

# 6.1.2 存储器分段管理

## ✓ 段寄存器引用示例一

在实方式下，某个段的段值为F000H，现要把段内最低的8个字节的内容送到两个32位的通用寄存器（EAX和EDX）中。

```
MOV    AX，0F000H
MOV    DS，AX          ;DS含段值F000H
MOV    ESI，0          ;使ESI为0（最低地址偏移为0）
MOV    EAX，[ESI]      ;取出最低的4个字节
MOV    EDX，[ESI+4]    ;再取出次低的4个字节
```

# 6.1.2 存储器分段管理

## ✓ 段寄存器引用示例二

在实方式下，现要求把位于F000H段开始处的32个字节的数据复制到开始地址为B800:2000H的区域。

```
MOV    AX, 0F000H      ;对应源段的段值
MOV    DS, AX          ;使DS含源数据段的段值
MOV    AX, 0B800H      ;对应目标段的段值
MOV    ES, AX          ;使ES含目标数据段的段值
MOV    ESI, 0          ;ESI=0
MOV    EDI, 2000H      ;EDI=2000H
MOV    ECX, 8          ;ECX=8，作为循环计数
```

实方式下，
F000H和B800H都是段值。

# 6.1.2 存储器分段管理

## ✓ 段寄存器引用示例二

在实方式下，现要求把位于F000H段开始处的32个字节的数据复制到开始地址为B800:2000H的区域。

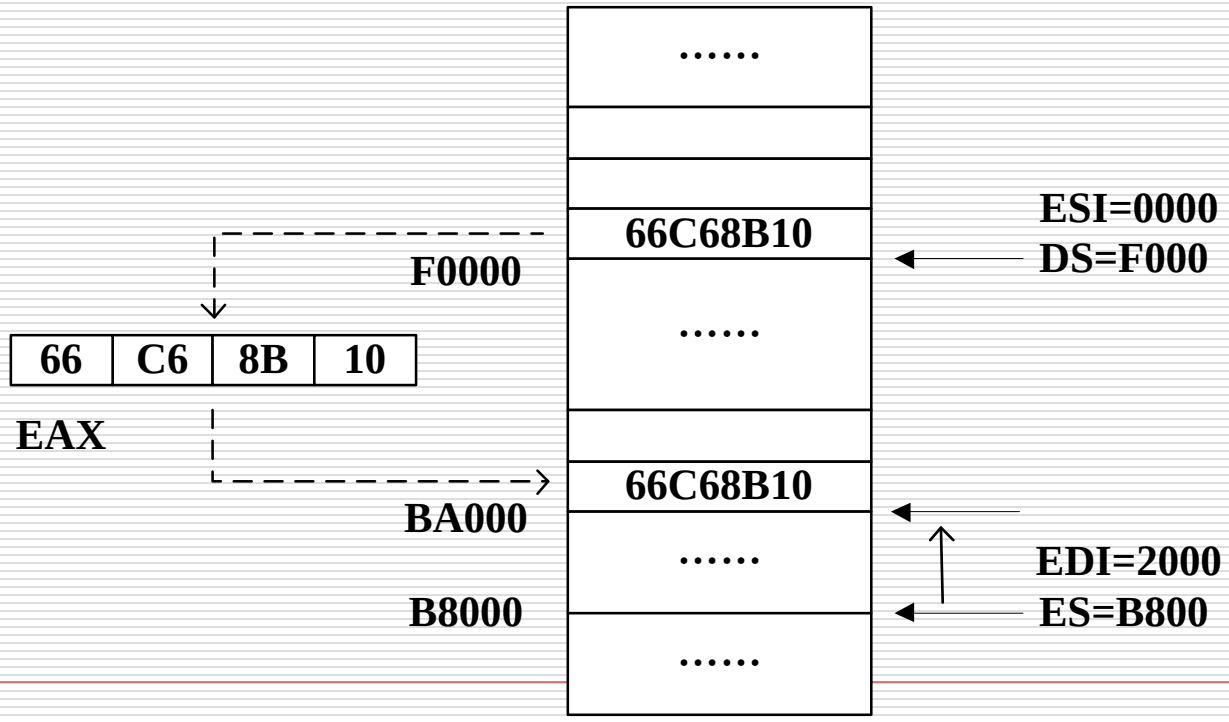
```
NEXT:
    MOV   EAX, [ESI]       ;自动引用DS，偏移为ESI值
    MOV   [ES:EDI], EAX    ;引用段寄存器ES，偏移为EDI值
    ADD   ESI, 4
    ADD   EDI, 4
    LOOP  NEXT
```

如何利用字符串操作指令？

# 6.1.2 存储器分段管理

## ✓ 段寄存器引用示例二

在实方式下，现要求把位于F000H段开始处的32个字节的数据复制到开始地址为B800:2000H的区域。



# 6.1.3 16位存储器寻址方式

## ➢16位存储器寻址方式

- ✓ 为了保持与早先处理器的兼容，IA-32系列处理器还支持16位的存储器寻址方式，也就是给出16位的存储单元有效地址，或者说给出16位的偏移。
- ✓ 16位的存储器寻址方式，主要应用于实方式。在实方式下，存储段的长度不超过64K，存储单元的有效地址是16位。

32位寻址方式同时有效！

# 6.1.3 16位存储器寻址方式

## ➢16位存储器寻址方式

✓ 16位有效地址EA可以有多种表示形式

基址　　变址　　位移量

$$EA = \begin{bmatrix} BX \\ BP \end{bmatrix} + \begin{bmatrix} SI \\ DI \end{bmatrix} + \begin{bmatrix} 8位 \\ 16位 \end{bmatrix}$$

基址部分可以是寄存器**BX**或**BP**；
变址部分可以是寄存器**SI**或**DI**；
位移量采用补码形式表示，在计算有效地址时，如位移量是**8**位，则被带符号扩展成**16**位。

# 6.1.3 16位存储器寻址方式

## ➢演示16位存储器寻址方式的使用

✓ 演示16位存储器寻址方式的使用

```
MOV    [DI], AX
ADD    DL, [SI+100H]
SUB    CX, [BX+DI-4]
MOV    [BX+SI+1230H], AL
MOV    DX, [BP+8]
```

操作数是8位或者16位

# 6.1.3 16位存储器寻址方式

## ➢演示16位存储器寻址方式的使用

✓ 演示16位存储器寻址方式的使用

```
MOV    EAX, [SI]
ADD    EDX, [DI-4]
SUB    [BX+DI], ECX
MOV    [BX+SI+3], EAX
```

操作数是32位

# 6.1.3 16位存储器寻址方式

## ➢演示16位存储器寻址方式的使用

✓ 如下指令中的16位存储器寻址方式的使用是非法的

```
MOV    EAX, [SI+DI]
MOV    DX, [AX]
MOV    [CX-3], AL
```

寻址方式非法！

采用32位寻址方式应该如何表示？

# 6.1.3 16位存储器寻址方式

## ➢演示16位存储器寻址方式的使用

✓ 演示针对16位存储器寻址方式的取有效地址指令

```
MOV    DI, 1234H              ;DI=1234H
MOV    BX, 16H                ;BX=0016H
LEA    SI, [DI+BX+5]          ;SI=124FH
LEA    EAX, [BX+DI-2]         ;EAX=00001248H
```